Sales prediction analysis using python

Load the dataset

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

data = pd.read_csv('/content/Advertising.csv')
```

Explore the data

+ Code + Text

```
print(data.head())
print(data.describe())
```

```
   Unnamed: 0     TV  Radio  Newspaper  Sales
0           1  230.1   37.8       69.2   22.1
1           2   44.5   39.3       45.1   10.4
2           3   17.2   45.9       69.3    9.3
3           4  151.5   41.3       58.5   18.5
4           5  180.8   10.8       58.4   12.9
       Unnamed: 0          TV       Radio   Newspaper       Sales
count  200.000000  200.000000  200.000000  200.000000  200.000000
mean   100.500000  147.042500   23.264000   30.554000   14.022500
std     57.879185   85.854236   14.846809   21.778621    5.217457
min      1.000000    0.700000    0.000000    0.300000    1.600000
25%     50.750000   74.375000    9.975000   12.750000   10.375000
50%    100.500000  149.750000   22.900000   25.750000   12.900000
75%    150.250000  218.825000   36.525000   45.100000   17.400000
max    200.000000  296.400000   49.600000  114.000000   27.000000
```

Separate features (X) and target variable (y)

```
X = data[['TV', 'Radio', 'Newspaper']]
y = data['Sales']
```

Split the data into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Create a linear regression model

```
model = LinearRegression()
```

Train the model

```
model.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

Make predictions on the test set

```
y_pred = model.predict(X_test)
```

Evaluate the model

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('Mean Squared Error:', mse)
print('R-squared:', r2)
```

```
Mean Squared Error: 3.1740973539761033
R-squared: 0.899438024100912
```

Print the coefficients and intercept

```
print('Coefficients:', model.coef_)
print('Intercept:', model.intercept_)
```

```
Coefficients: [0.04472952 0.18919505 0.00276111]
Intercept: 2.979067338122629
```

Analyze feature importance

```
importance = pd.DataFrame({'Feature': X.columns, 'Coefficient': model.coef_})
print(importance)
```

```
      Feature  Coefficient
0          TV     0.044730
1       Radio     0.189195
2   Newspaper     0.002761
```

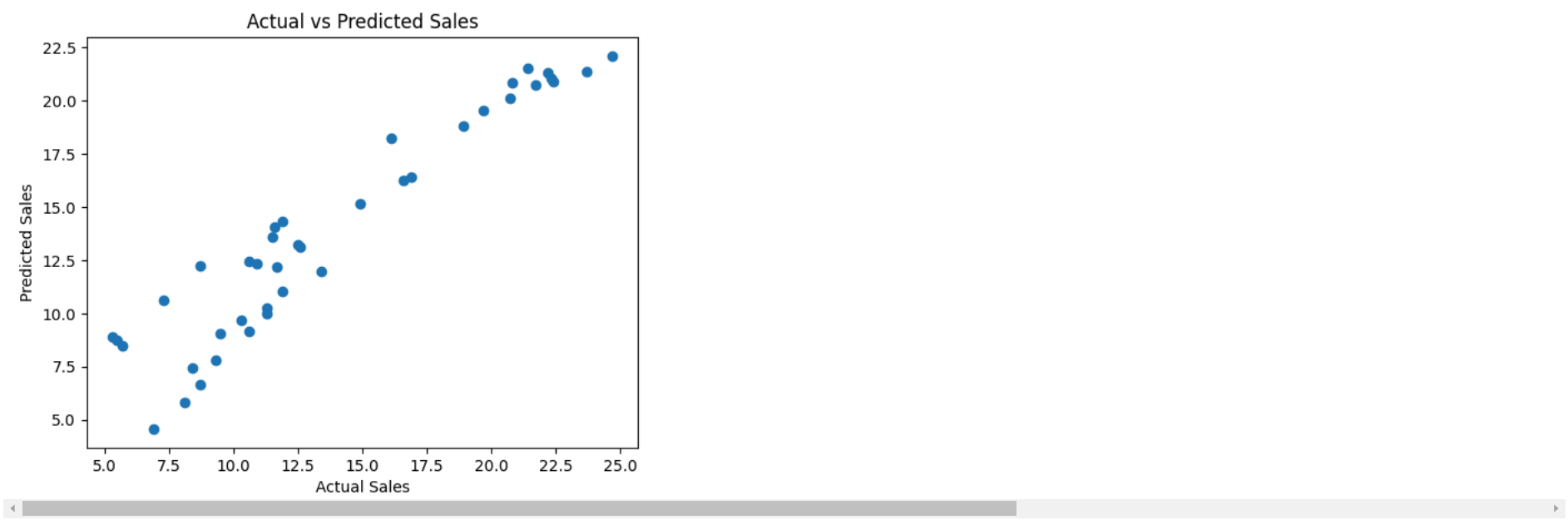
Make predictions on new data

```
new_data = pd.DataFrame({'TV': [100], 'Radio': [50], 'Newspaper': [25]})
prediction = model.predict(new_data)
print('Prediction for new data:', prediction)
```

```
Prediction for new data: [16.98079966]
```

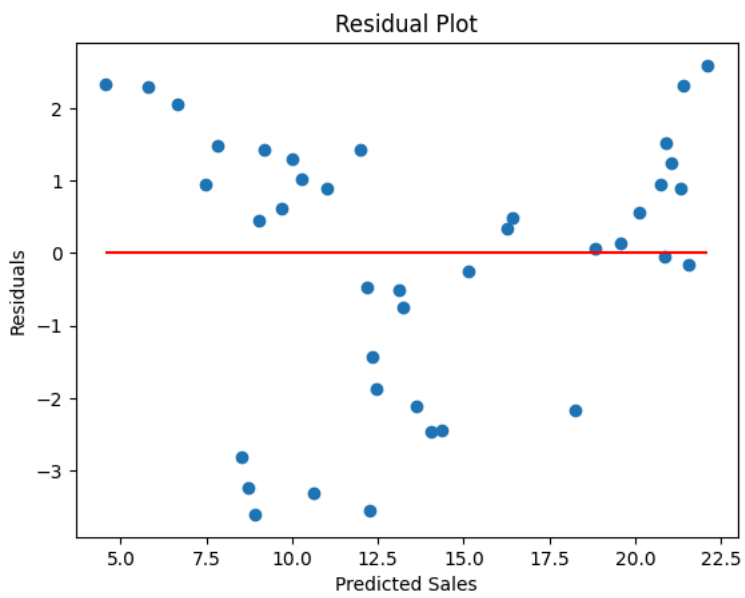
Visualize the results

```
import matplotlib.pyplot as plt
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs Predicted Sales')
plt.show()
```



Examine the difference between actual and predicted values to identify patterns.

```
residuals = y_test - y_pred
plt.scatter(y_pred, residuals)
plt.xlabel('Predicted Sales')
plt.ylabel('Residuals')
plt.title('Residual Plot')
plt.hlines(y=0, xmin=y_pred.min(), xmax=y_pred.max(), color='red')
plt.show()
```



Analyze the correlation between features and sales.

```
correlation_matrix = data.corr()
print(correlation_matrix)
```

```
             Unnamed: 0        TV     Radio  Newspaper     Sales
Unnamed: 0    1.000000  0.017715 -0.110680  -0.154944 -0.051616
TV            0.017715  1.000000  0.054809   0.056648  0.782224
Radio        -0.110680  0.054809  1.000000   0.354104  0.576223
Newspaper    -0.154944  0.056648  0.354104   1.000000  0.228299
Sales        -0.051616  0.782224  0.576223   0.228299  1.000000
```

Feature Engineering

```
data['TV_Radio_Interaction'] = data['TV'] * data['Radio']
X = data[['TV', 'Radio', 'Newspaper', 'TV_Radio_Interaction']]
y = data['Sales']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
```

```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('Mean Squared Error (with interaction term):', mse)
print('R-squared (with interaction term):', r2)
```

```
Mean Squared Error (with interaction term): 0.8144305830812308
R-squared (with interaction term): 0.9741971529119294
```

Polynomial Regression

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('Mean Squared Error (with polynomial features):', mse)
print('R-squared (with polynomial features):', r2)
```

```
Mean Squared Error (with polynomial features): 0.47178944590714467
R-squared (with polynomial features): 0.9850527335497992
```

Cross-Validation

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(model, X_poly, y, cv=5, scoring='neg_mean_squared_error')
print('Cross-Validation Scores:', -scores)
print('Average MSE:', -scores.mean())
```

```
Cross-Validation Scores: [0.32930934 0.31759268 0.19116681 1.25029619 0.16651293]
Average MSE: 0.45097559041534485
```

Regularization

```
from sklearn.linear_model import Ridge

model = Ridge(alpha=1.0)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

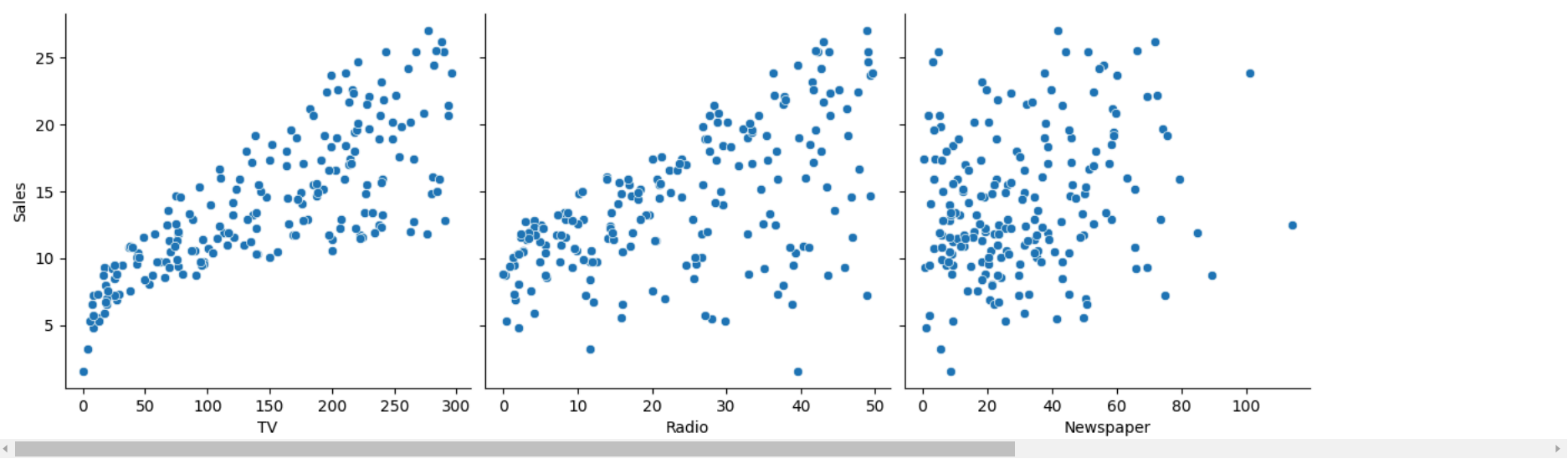
```
print('Mean Squared Error (with Ridge Regression):', mse)
print('R-squared (with Ridge Regression):', r2)
```

```
Mean Squared Error (with Ridge Regression): 0.47158182800077053
R-squared (with Ridge Regression): 0.98505931131493
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_ridge.py:200: LinAlgWarning: Ill-conditioned matrix (rcond=4.46191e-18): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
```

Pairplot to visualize relationships between all variables

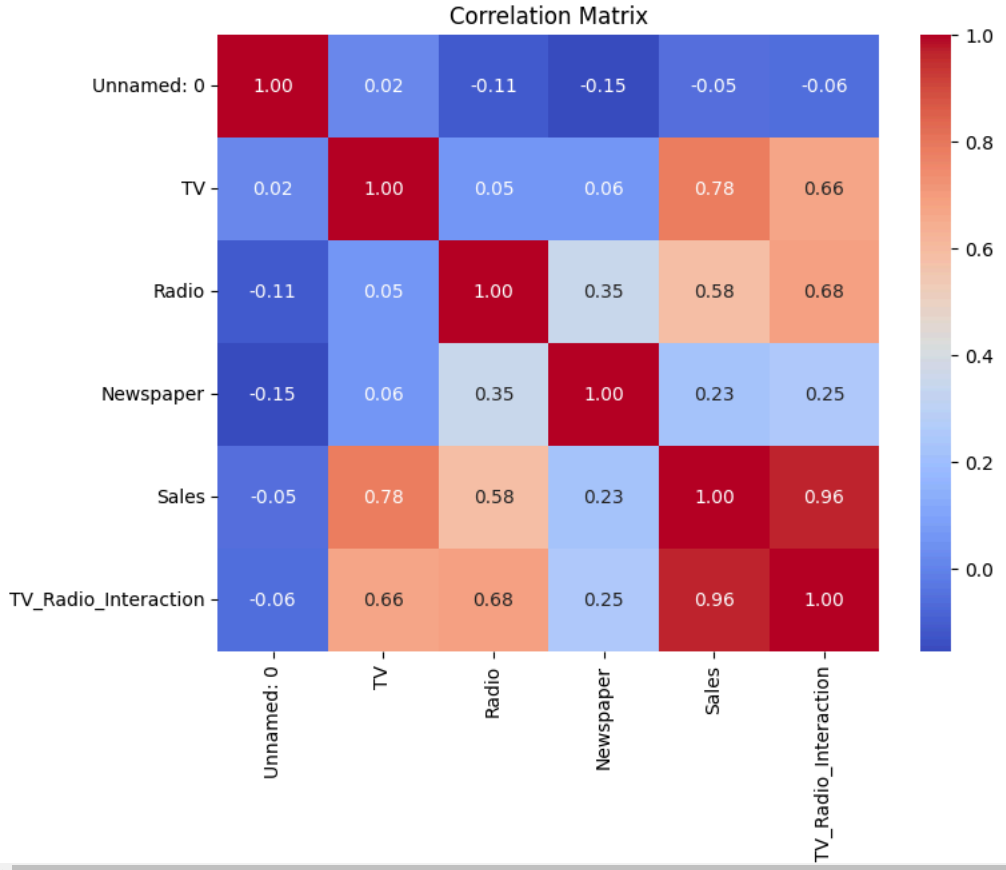
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.pairplot(data, x_vars=['TV', 'Radio', 'Newspaper'], y_vars='Sales', height=4, aspect=1)
plt.show()
```



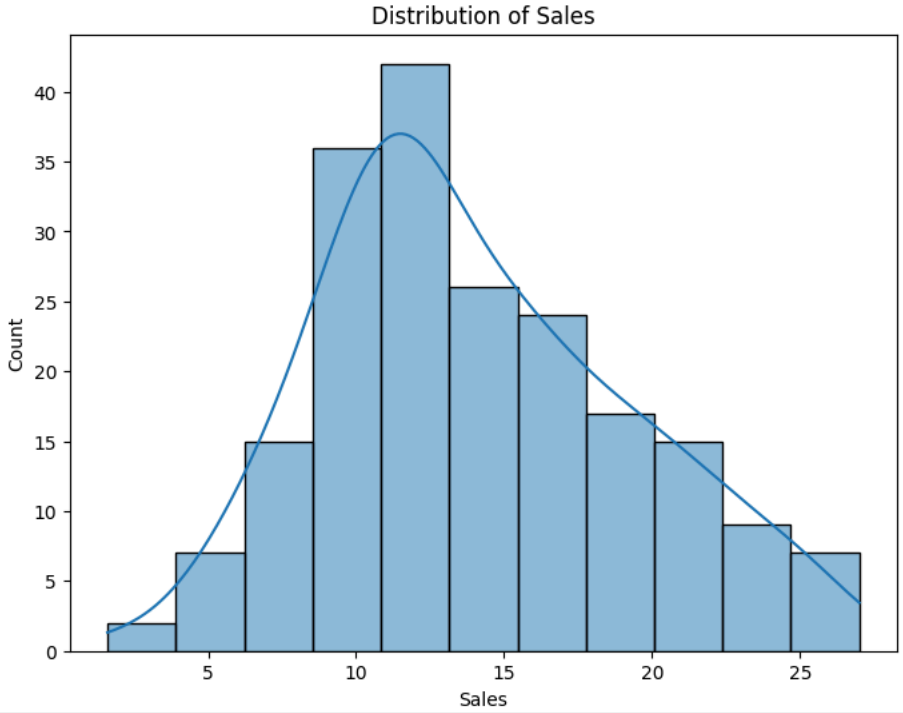
Correlation heatmap

```
plt.figure(figsize=(8, 6))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



Distribution of Sales

```
plt.figure(figsize=(8, 6))
sns.histplot(data['Sales'], kde=True)
plt.title('Distribution of Sales')
plt.show()
```

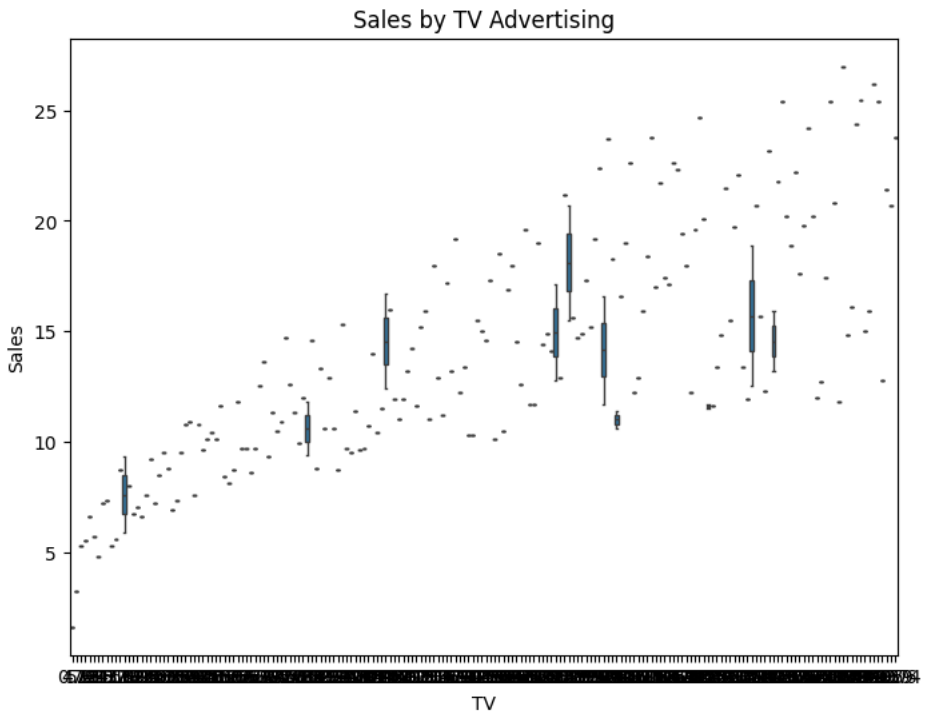


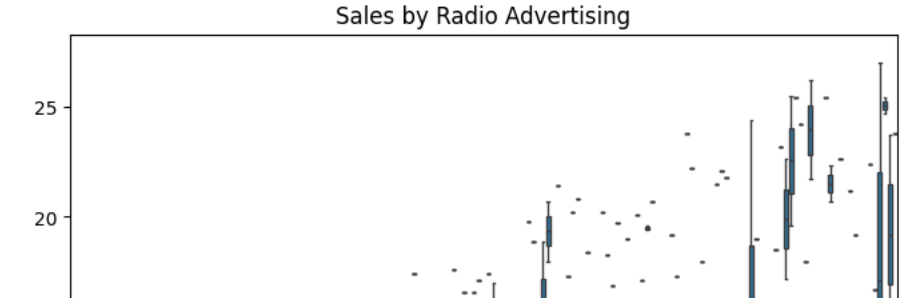
Boxplot of Sales by different advertising media

```
plt.figure(figsize=(8, 6))
sns.boxplot(data=data, x='TV', y='Sales')
plt.title('Sales by TV Advertising')
plt.show()

plt.figure(figsize=(8, 6))
sns.boxplot(data=data, x='Radio', y='Sales')
plt.title('Sales by Radio Advertising')
plt.show()

plt.figure(figsize=(8, 6))
sns.boxplot(data=data, x='Newspaper', y='Sales')
plt.title('Sales by Newspaper Advertising')
plt.show()
```


Sales by TV Advertising
Sales
TV
25
20
15
10
5


Sales by Radio Advertising
25
20